Application Note #128

SERIAL PORT OVERRUNS

(The following is an article from Computer Telephony magazine, March 1995, which answers commonly asked questions.)

Modems are getting faster. Serial ports are not. Bingo, serial port overruns and communications problems. There are explanations and solutions. Here they are from an expert in PC communications.

by Pete Maclean

Overrun refers to the condition that occurs when a computer cannot keep pace with the data arriving at a serial port. Imagine characters arriving on a conveyor belt with a small holding area at the end and a worker who unloads them. If the worker cannot keep up with the arrival late or if he gets distracted for a moment, then the holding area may overflow and some characters fall off and get lost That's an overrun.

A maddening example of the same phenomenon is when you are riding a packed escalator and the person in front of you gets off and immediately stops to look around. You have no choice but to bump into him because there's nowhere else to go.

This article explains overruns in detail and suggests strategies for avoiding them overruns can, in principle, occur on any kind of computer but our discussion is mainly for PC users who suffer particularly from them.

Here's a fuller explanation of an overrun: Each time a new character arrives at a COM port, it is held in a register in the port's hardware until the PC processor takes it away. To summon the processor, the port sends it an interrupt. Such an interrupt is a hardware signal that, under suitable conditions, forces the processor to set aside whatever it's doing and briefly switch control to an interrupt service routine that does what is needed to service the device that generated the interrupt. In the ease in question. The interrupt service routine reads the character from the port and, typically, stores it in memo.

An overrun occurs when a second character arrives at the port before the first one has been unloaded. The port has nowhere else to stick the second character so it stores it in the same register. Overwriting the first. So the cause of an overrun is congestion and the effect is that a character gets lost.

WHY DO OVERRUNS HAPPEN?

Overruns have become a nuisance of late because, while most PC components have over time increased in speed by a factor of between 10 and 100, serial ports have not. (Diskette drives have also not increased much in speed but this has not given rise to significant problems.) Modems, in particular. can handle traffic rates about 50 times higher than at the start of PC history'. However, the best standard serial ports have performance only 16 times better and many PCs are still built with ports that don't incorporate this improvement.

Two primary factors combine to increase the chance of overruns. First, the processor may not be able to respond to an interrupt immediately. This can be either because it is busy servicing an interrupt of higher priority or because the software that the processor happens to be executing has interrupts inhibited.

In PCs, like most computers, interrupts are prioritized so that those considered most important are serviced first while others are made to wait their turn. The PC architecture always gives the clock and the keyboard higher priority than COM ports and a particular configuration may allocate higher interrupt priorities to other devices as well. Interrupt service routines generally complete their work very quickly but if there are enough that have to complete before the COM port gets its service then sufficient time can pass for an overrun to occur.

Many types of software need to turn off interrupts occasionally for brief periods. Device drivers, task switchers, DOS TSRs, and disk caching utilities are common examples. Every time this is done, albeit that there's a legitimate need to do so, the chance of an overrun is increased.

There's a third software issue that should be mentioned. In a case with a slow PC receiving data at a high rate, say a 4.77MHz XT and a 19,200bps line, even if other devices are inactive and the processor is always available to service COM port interrupts, an overrun may occur simply because the time it takes the processor to execute all the instructions in the interrupt service routine is longer than the inter-character arrival time.

The second of the two factors mentioned above is the COM port hardware. The simplest comport hardware has room to hold only one incoming character. Since memory Is cheap you may well ask why a port does not have a bunch of RAM to hold multiple characters. The answer is that some COM ports, in fact, do, and one solution to the overrun problem may be to upgrade your serial port hardware as described below. But the critical issue is that too darned many PC COM ports are built using cheap obsolete hardware that has but

one holding register for incoming data.

One of several common misconceptions about overruns is that they are caused by software. It should be clear to you now that overruns are in fact hardware errors,' communications software may dutifully report such errors but is typically powerless to prevent them.

Another frequently met misconception is that flow control can offer a solution to overruns. In fact, flow control is designed to regulate the flow of characters at a coarse level to avoid overflowing data buffers managed by software. When a PC sends a flow control signal to suspend the incoming traffic it may be several character times before the effect is seen. Flow control is not designed to, and in practice simply cannot, regulate flow at the fine level that would be necessary to avoid overruns. Never mind the fact that flow control has to be managed by software PC serial ports do not have that capability and if software cannot get control quickly enough to unload the port then it also could not get control quickly enough to exert flow control even if that could be effective!

HOW CAN OVERRUNS BE AVOIDED?

There are five possible approaches to avoiding overrun errors:

1. **Upgrade your hardware.**

   Getting a faster PC, assuming you don't already have the fastest on the market, will reduce the risk of overruns because the processor can run interrupt service routines more quickly and so keep the COM port waiting for shorter periods. Upgrading your entire system is, however, not usually an economical solution and it's typically not one that can guarantee to reduce all risk of overruns anyway.

   If you are in a position to consider a more modest hardware investment then consider upgrading your serial port. Users of portable PCs can switch to a PCMCIA modem from a serial port one. Thus your new modem serial port will be what's in the PCMCIA modem. In the newer ones, you often find an upgraded "serial port."

   The simplest possible upgrade is to change a chip in your existing port. The heart of any COM port is a chip called a UART (Universal Asynchronous Receiver Transmitter). This is the chip that holds the incoming characters until the processor can take them. You can determine what type of UART your COM ports have by running one of several software utilities designed for the purpose. One candidate is Microsoft's MSD. If you find out that your port is built using an UART identified as an 8250 or 16450 then it's one of those dinosaurs that has only room for one character. If that chip is scooted you can replace it with a 16550A which should cost about $10. The 16550A is the latest and greatest PC compatible UART and can hold 16 inbound characters!

   The 16550A is a great advance over earlier UARTs but it's still far from ideal. For one thing, while capable of storing 16 inbound characters, it only actually does so when software tells it to. By default it acts like a dinosaur and stores just one!

   Many communications packages these days recognize a 16550A and initialize it to use its 16 character memory but, be warned, some do not. The other consideration about the 16550A is that, in extreme cases. even the elasticity permitted by a 16character buffer may not be sufficient to totally avoid overruns. In most practical cases it is; but there are no guarantees.

   A slightly more expensive step is to replace an entire port. Serial port cards equipped with 16550As are commonly available and should cost around $35. Especially smart serial ports are available from Digiboard, Hayes, Multitech and Telcor. They are more expensive but often a worthwhile investment, especially if you run Microsoft Windows. We use and like the Hayes ESP (Enhanced Serial Port) which lists at $99. and also hear good reports of the Telcor T/Port Adapter. (Hayes Microcomputer, Atlanta, GA 408-840-9200 Telcor Systems, Natick, MA 800-826-2938.)

   Many magazines are now noting in comparative reviews of notebook PCs which units are built with 16550Abased COM ports. Sometimes these are flagged as simply having "high speed" COM ports. We recommend buying only notebooks that are so equipped.

   The advent of high speed PCMCIA modems has proved to be a blessing for notebook users. Such modems have COM ports built into them and, of necessity, these are designed as 16550A compatible. So, should your notebook manufacturer have decided to save a couple of bucks on every unit by installing cheaper UARTs for the COM port(s), you need not be affected, at least for modeming.

2. **Reduce the COM port speed.**

   Short of upgrading the hardware, reducing the speed, say from 9,600 to 4,800 or even 2,400, is the surest method to avoid overruns. You may consider this only a last resort but at least it's a recourse you can depend on.

3. **Upgrade your software.**

   As mentioned above, even if you have a 1655OA equipped port, you still need software that recognizes that and takes advantage of it. If you use an advanced operating system such as NextStep, NT, OS/2 or UNIX then that job falls to the serial port driver in the system itself and, in all cases we know of, you can depend on it's being done. If you use DOS, then

responsibility falls to each application. If you use windows then the situation gets rather complicated; this will be covered later.

4. **Use a protocol.**

   When software detects an overrun it can do one of three things. First, it can ignore the condition. It is not uncommon for terminal emulators to do so, which means that you may observe characters missing from your output. Second, the software can report the condition to you. Third. the software can try to recover from the loss; but only if it is operating a protocol that allows such a move.

   When a program is using ZMODEM, say, to transfer a file it may well ignore overruns per se because it can depend on an overrun giving rise to a block check (CRC) error, just as it would were data corrupted by any other fault. The damaged data block is simply retransmitted until it gets through okay. The trouble with overruns may be that they occur often enough that many blocks have to be retransmitted and performance degrades. In a pathological case, one could see better file transfer performance at, say, 9600 bps than at 19200.

   Modern modems support error correction by operating a protocol such as MNP or V.42. This means that your data is guaranteed reliable passage over most of its journey and is vulnerable to damage only on its way between your modem and software. While other problems can occur here. overruns are the only errors likely to impinge. In a few cases, a remedy is to use software based MNP rather than a modem based protocol.

   Using your modem's MNP is usually preferable in terms of performance. If you get overruns, however, and your software can do MNP then you may need to trade some performance for reliability. One software package that can operate MNP is MCI Mail Express for PCs, a DOS front-end for MCI Mail.

   It's the user's choice: one can configure the program to use its MNP driver or not. A user who lacks an MNP capable modem should certainly take advantage of the software MNP; but even with an error correcting modem, one may properly prefer to use software MNP. It may be necessary to determine by experiment whether you get better performance operating modem based MNP at a low speed or software based MNP at a high speed. By the way, some people who should know better have published statements that it is never right to use software MNP if one has a modem that is MNP capable. You should now appreciate why that claim is wrong.

   Switching on software MNP rather than using your modem for error correction does not actually eliminate overruns but allows the software to recover from them in the manner described above for ZMODEM. When switching to using Software MNP, bear in mind that the modem must be configured to not operate MNP.

5. **Eliminate the Culprit.**

   It is rare but sometimes the case that, when overruns are being caused by the action of some software keeping interrupts off too long, you can modify or eliminate that software. Disk caching programs for DOS, especially those that do write caching, may be such culprits.

SPECIAL CONSIDERATIONS WITH WINDOWS

Overrun errors occur more frequently in Windows than in plain DOS systems. The main reason is that Windows itself turns interrupts off for substantial periods while task switching. When considering overruns and windows, one must consider native Windows applications and DOS programs running in Windows DOS VMs separately.

WINDOWS APPLICATIONS

Fortunately, the Windows 3.1 COM port driver (COMM.DRV) has the capability of enabling the FIFOs (those 16character buffers) in 16550Aequipped ports. Unfortunately it does so only when told to! The operative cue is the COMnFIFO parameter in the [386Enh] section of the SYSTEM.INI file. This should be set to 1. as in this example:

COMiFIFO = 1

Be careful, some documentation tells you to use the form:

COMiFIFO = ON

but this does not work!

Unfortunately the driver is hardwired to set the interrupt threshold to 14. This means that the COM port generates an interrupt only once it has accumulated 14 characters, leaving but two character times (i.e. while the 15th and 16th characters arrive) for the processor to respond. The hardware supports thresholds of 2,4, and 8 as well, and 8 would have been a much better choice. Still, even with the threshold of 14, having the FIFOs enabled does make a big difference.

We wonder why COMM.DRV demands that COMn FIFO be set. We cannot see any danger in just always er abling the FIFOs when they are present

Of course, if you do not have suitable COM ports then this facility of Windows cannot help you. It is our experience that, without FIFOs, it is rare that communication at speeds of 9,600 bps and higher can be sustained without overruns. One does, however, have all the options listed above for improving matters. When working with Windows. we particularly recommend considering use of software MNP when available.

DOS APPLICATIONS UNDER WINDOWS

The Windows COMM.DRV mentioned above serves Windows applications and Windows applications only. DOS applications running within the Windows environment are served by another COM port driver called CommBuff. This serves to partially virtualize COM ports for DOS applications. Unfortunately, CommBuff is not as clever as COMM.DRV it does not support 16550A FIFOs and thus makes all COM ports appear to DOS programs as "dinosaur" COM ports.

Therefore. when running under Enhanced Mode Windows, a DOS application cannot take advantage of a COM port's FIFOs even when it knows how.! In this situation, upgrading the COM port hardware does not help and one must often fall back on either software MNP or reducing speed to avoid overruns.

An alternative is to use third party replacement COM port software called TurboCom. This product provides replacements for both COMM.DRV and CommBuff, the latter with a driver that does support FIFOs. TurboCom is available from Pacific CommWare, Ashland, OR 5034822744: CIS 71521,760 and MCI 3445374. While we like and use TurboCom's CommBuff replacement. we are not fans of its alternative COMM.DRV as it seems to cause problems for some Windows applications.